Ibmec

Comunicação interaplicações WEB

Webservices, SOAP, REST, XML, JSON, WebSockets

Prof. Eduardo Mangeli

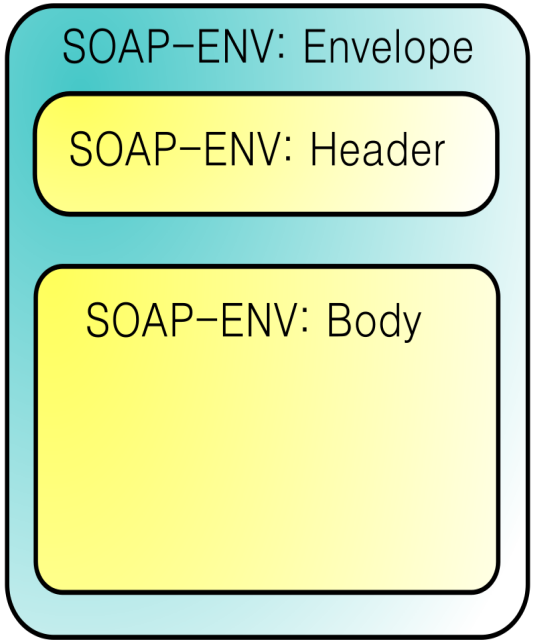
SOAP

Características Gerais

- Protocolo de comunicação entre serviços
- Permite invocar processos rodando em diferentes computadores
- Independente de linguagem ou sitema operacional
- Mensagens podem ser transportadas por SMTP, HTTP, HTTPS e mesmo utiliando UDP

Estrutura de uma mensagem

Elemento	Description	Obrigatório
Envelope	Identifica o documento XML como uma mensagem SOAP	Sim
Header	Informações de cabeçalho	Não
Body	Chamadas e respostas	Sim
Fault	Informações sobre erros	Não



Exemplo de Mensagem - 1

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body>
  <m:GetPrice xmlns:m="https://www.w3schools.com/prices">
    <m:Item>Apples</m:Item>
  </m:GetPrice>
</soap:Body>

</soap:Envelope>
```

Exemplo de Mensagem - 2

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body>
  <m:GetPriceResponse xmlns:m="https://www.w3schools.com/prices">
    <m:Price>1.90</m:Price>
  </m:GetPriceResponse>
</soap:Body>

</soap:Envelope>
```

Crítica

- Complexidade da implementação
- Sem modelo padronizado
- Interpretação custosa.

REST

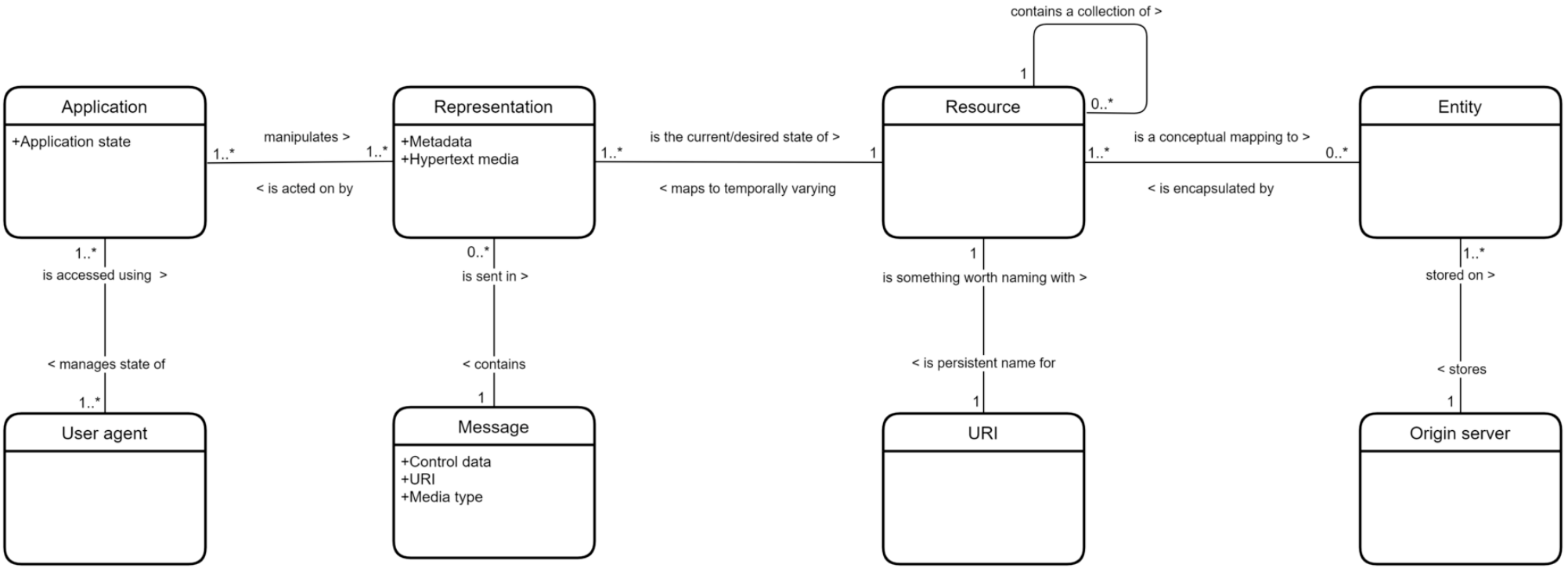
Representational State Transfer

Características Gerais

- Estilo arquitetural
- Descreve uma *interface* entre sistemas
- Modelo Cliente-Servidor

Restrições Arquiteturais

- **Cliente-Servidor**
  - separação de responsabilidades
- **Sem Estado**
  - não existe informação retida
- **Uso de Cache**
  - acelerar o funcionamento da rede
- **Sistema em Camadas**
  - permite o uso de *proxies* e balanceadores
- **Interface Uniforme**
  - identificação do recurso na requisição
  - manipulação de recursos por representação
  - mensagens auto descritivas
  - links para recursos de mídia nas mensagens


contains a collection of >
1
0..*
Application
+Application state
manipulates >
1..*
1..*
< is acted on by
Representation
+Metadata
+Hypertext media
1..*
is the current/desired state of >
1
< maps to temporally varying
Resource
1..*
is a conceptual mapping to >
0..*
< is encapsulated by
Entity
1..*
is accessed using >
< manages state of
1..*
User agent
0..*
is sent in >
< contains
1
Message
+Control data
+URI
+Media type
1
is something worth naming with >
< is persistent name for
1
URI
1..*
stored on >
< stores
1
Origin server

Métodos HTTP aplicados ao REST

Método	Descrição
GET	Obtém a representação do estado de um recurso
POST	Solicita que o recurso processe e representação enviada
PUT	Cria ou substitui o estado do recurso com a representação enviada
PATCH	Atualiza (parcialmente) o recurso
DELETE	Apaga o estado do recurso
OPTIONS	Obtém métodos disponíveis

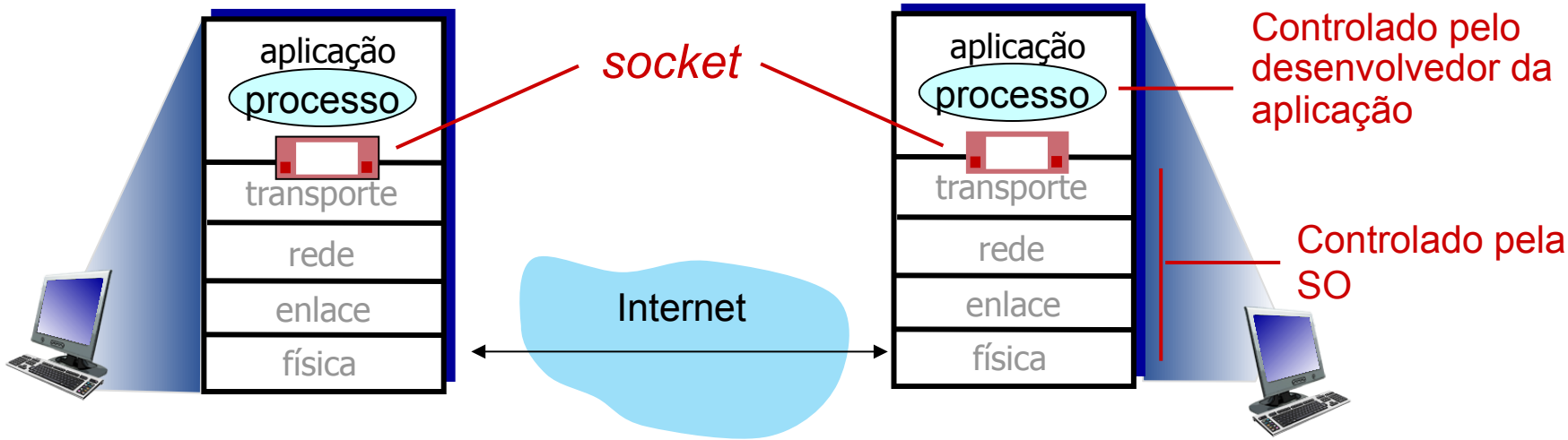
- APIs de que são totalmente aderentes às restrições arquiteturais do REST são chamadas de RESTful
- REST é um padrão arquitetural e SOAP é um protocolo de comunicação
- REST não é um padrão

Sockets

Programando sockets

objetivo: aprender como construir uma aplicações cliente/servidor que se comunicacam usando sockets

socket: portas entre processos de aplicações e protocolos de transporte



Dois tipos

- UDP: não confiável, datagrama
- TCP: confiável, orientação a conexão

Socket UDP

UDP: sem estebelecimento de “conexão” entre cliente e servidor

- sem *handshaking* antes de enviar os dados
- IP e número da porta adicionados a cada pacote enviado
- receptor extrai IP de quem envia número da porta do pacote recebido

UDP: dados transmitidos podem ser perdidos ou recebidos foram de ordem

Ponto de vista da aplicação:

- UDP provê transporte *não-confiável* para grupos de bytes (“datagramas”) entre cliente e servidor

Exemplo cliente UDP

```
from socket import socket, AF_INET, SOCK_DGRAM

server_name = 'localhost'
server_port = 12000

client_socket = socket(AF_INET, SOCK_DGRAM)

msg = input('Entre minúsculas \n')

client_socket.sendto(msg.encode(), (server_name, server_port))

msg_modificada, server_adddress = client_socket.recvfrom(2048)

print(msg_modificada.decode())

client_socket.close()
```

Exemplo de Servidor UDP

```
from socket import socket, AF_INET, SOCK_DGRAM

server_port = 12000

server_socket = socket(AF_INET, SOCK_DGRAM)

server_socket.bind(('', server_port))

print("Servidor pronto!")

while True:
    msg, client_address = server_socket.recvfrom(2048)
    msg_modificada = msg.decode().upper()
    server_socket.sendto(msg_modificada.encode(), client_address)
```

Socket TCP

- Cliente precisa contactar o servidor;
  - servidor precisa ter um socket para receber o contato
- Quando o cliente estabelece a conexão como servidor, ele cria um socket;
- O servidor cria um socket ao estabelecer a conexão
  - número da porta usado para distinguir clientes
  - permite se comunicar com múltiplos clientes

- Socket TCP provê transporte confiável, em orderm, através de um streaming entre cliente e servidor

Exemplo Servidor TCP

```
from socket import socket, AF_INET, SOCK_STREAM

server_port = 12000

server_socket = socket(AF_INET, SOCK_STREAM)

server_socket.bind(('', server_port))

server_socket.listen(1)

print("Servidor pronto!")

while True:
    connection_socket, addr = server_socket.accept()

    msg = connection_socket.recv(1024).decode()
    msg_modificada = msg.upper()
    connection_socket.send(msg_modificada.encode())

    connection_socket.close()
```

Exemplo Cliente TCP

```
from socket import socket, AF_INET, SOCK_STREAM

server_name = 'localhost'
server_port = 12000

client_socket = socket(AF_INET, SOCK_STREAM)

client_socket.connect((server_name, server_port))

msg = input('Entre minúsculas \n')

client_socket.send(msg.encode())

msg_modificada = client_socket.recv(1024)

print(msg_modificada.decode())

client_socket.close()
```

Vamos praticar !